OIPE IAP48 OCT 05 2006 PATENT & TRADEMARK OFFICE

AMENDMENTS TO THE SPECIFICATION

In response to the objections presented in section 6 of the Office Action, replace the following paragraphs as indicated:

Abstract:

Disclosed is a compiler apparatus for generating an instruction code composed of instruction sets each including an instruction that designates an m-bit immediate value indicating a location of a data item in a memory area. The compiler apparatus sequentially selects, based on one data attribute, a data item from a group X composed of a plurality of data items; and judges, each time a data item is selected, whether the selected data item is allocatable to an n-byte memory area ($n \leq 2^m$). When the judgment is negative, the compiler apparatus specifies, based on a different data attribute, a data item out of all the selected data items and [[exclude]] excludes the specified data item from the group X, and repeats the selection until all the data items remaining in the group X after excluding all the specified data items are judged to be allocatable to the memory area.

Page 16:

Here, there are three variables *a*, *b* and *c* remaining in the target-variable set, so that the CPU 101 judges accordingly (step S1012: N), and selects the variable *b* having a largest alignment value in the target-variable set (step S1005). The CPU 101 then determines to allocate the variable [[*d*]] *b* to a location in the stack area whose address begins at 32 (step S1006). This is because the variable *b* with the alignment 4 must be allocated to a location in the stack area that (i) has not been determined as a location for any variable, (ii) satisfies the alignment

constraint, and (iii) has a smallest possible address. As shown in FIG. 5, the variable [[*d*]] *b* has been already determined to be allocated in the stack area to occupy the location that begins at the address 0 and ends at the address 31. Thus, the variable [[*d*]] *b* must be allocated to a location whose address is a multiple of 4 that is equal to 32 or greater. Next, the CPU 101 removes the variable b from the target-variable set (step S1007), adds the variable *b* to the determined-variable set (S1008), and judges whether the address of the determined location falls within the address range of the address 0 to the address 31 (step S1009). FIG. 6 is a schematic view showing the locations of the variables *a* and *b* in the stack area at this stage.

Pages 17-18:

With the above processing, the variables *b*, *c* and *a* are determined to be allocated to locations in the stack area so as to [[be]] fall within a range that begins at the address 0 and ends at the address 23, as shown in a schematic view of FIG. 7. Accordingly, the CPU 101 judges that each determined location is within the predetermined address range (step S1009: Y), and further judges that there is no more variable remains in the target-variable set (step S1012: Y). Since the variable *d* is now in the exclusion-variable set, the CPU 101 judges accordingly (step S1013: Y). The CPU 101 then selects the variable *d* that is a smallest-size variable in the exclusion-variable set (to be more specific, the variable *d* is the only member of the exclusion variable set in this example). The CPU 101 determines to allocate the variable d to the location whose address is 24 (step S1015). This is because the variable *d* having the alignment 8 needs to be allocated to a location in the stack area that (i) has not been determined as a location for any variable, (ii) satisfies the alignment constraint (a location whose address is a multiple of 8), and (iii) has a smallest possible address (a location whose address is equal to 24 or greater). Next, the CPU 101

removes the variable *d* from the exclusion-variable set (step S1016), judges that no more variable remains in the exclusion-variable set (step S1013: N), and terminates the location determination processing.